**Εργασία Ανάκτησης Πληροφορίας**

**Δημιουργία μηχανής αναζήτησης**

**Σχολή Μηχανικών**

**Τμήμα Πληροφορικής και Υπολογιστών**



Σπυριδούλα Μήτσιου ΑΜ:21390133

Κωνσταντίνος Τσακίρης ΑΜ:21390224

https://github.com/SpyridoulaM/information\_retrieval

**Βήμα 1. Συλλογή δεδομένων:**

Για τοσύνολο δεδομένων σε αυτό το κομμάτι επιλέξαμε να χρησιμοποιήσουμε ένα έτοιμο από την ιστοσελίδα Kaggle. Συγκεκριμένα χρησιμοποιήσαμε το εξής [*dataset*](https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots/code) το οποίο περιέχει δεδομένα τα οποία είναι συλλεγμένα μέσω scraping από την Βικιπαίδεια. Scraping είναι η τεχνική με την οποία συλλέγονται δεδομένα από πηγές στις οποίες το περιεχόμενο είναι σε μορφή αναγνώσιμη από άνθρωπο και όχι μηχανή. Αυτή η τεχνική εξάγει τις πληροφορίες από τις επιλεγμένες ιστοσελίδες και μετατρέπει το περιεχόμενο σε δομή δεδομένων έτσι ώστε να μπορεί να χρησιμοποιηθεί μέσω κώδικα.

Το dataset περιέχει τα εξής στοιχεία για κάθε εγγραφή:

* Έτος κυκλοφορίας
* Τίτλος ταινίας
* Προέλευση
* Σκηνοθέτης
* Κύριοι ηθοποιοί
* Είδος ταινίας
* URL του άρθρου Βικιπαίδεια της ταινίας
* Πλοκή της ταινίας

Η επιλογή του έτοιμου dataset έγινε μετά από προσπάθεια scraping μέσω δικού μας κώδικα σε python, το οποίο έγινε χρονοβόρο στην γραφή του κώδικα , όπως και στην εκτέλεση του κώδικα, για να εξάγει αυτές τις πληροφορίες αυτόματα μέσω του ίντερνετ θέλει χρόνο και καλή σύνδεση στο ίντερνετ, στα οποία δεν είχαμε πρόσβαση. Το συγκεκριμένο dataset επιλέχθηκε λόγω του όγκου δεδομένων τα οποία περίεχει και της ευρείας χρήσης του από επιστήμονες στον τομέα.

Το dataset περιέχει εγγραφές για 34,886 ταινίες, για το οποίο οι υπολογιστικοί πόροι των συστημάτων μας δεν αρκούν, οπότε αποφασίσαμε να χρησιμοποιήσουμε τις 1000 πρώτες εγγραφές για την υλοποίηση της μηχανής αναζήτησης.

**Βήμα 2. Προεπεξεργασία κειμένου (Text Processing):**

Η προεπεξεργασία του κειμένου αποτελείται από τις εξής διαδικασίες:

1.Αφαίρεση ειδικών χαρακτήρων.

Από το κείμενο αφαιρούνται όλοι οι ειδικοί χαρακτήρες όπως το θαυμαστικό, το κόμμα και η τελεία.

2.Μετατροπή σε μικρά.

Όλα τα κεφαλαία γράμματα μετατρέπονται σε μικρά, με αυτό τον τρόπο η λέξη “Serve” και η λέξη “serve” έχουν το ίδιο αποτέλεσμα.

3.Tokenization

Tokenization είναι η τεχνική από την οποία η κάθε λέξη μετατρέπεται σε tokens, τα οποία ύστερα θα υποστούν περαιτέρω επεξεργασία πριν να είναι έτοιμα για χρήση.

4.Αφαίρεση stopwords

Αφαιρούνται όλες οι λέξεις οι οποίες είναι κατηγοριοποιημένες ως stopwords, δηλαδή λέξεις οι οποίες χρησιμοποιούνται συνεχώς στην γλώσσα και δεν έχουν κάποια εννοιολογική αξία (π.χ. “and”, “the” κ.ο.κ.).

5.Stemming

Λέξεις οι οποίες παράγονται από άλλες μετατρέπονται στην ρίζα τους ως tokens ώστε να αναγνωρίζονται οι σωστές λέξεις κατά την αναζήτηση, για παράδειγμα το query “help” και το query “helps” εμφανίζουν τα ίδια αποτελέσματα εξαιτίας αυτής της διαδικασίας.

6.Αφαίρεση πολλαπλών κενών

Ύστερα αφαιρούνται τα κενά από τα δεδομένα τα οποία χρησιμοποιούμε με αποτέλεσμα να υπάρχει μόνο ένα κενό ανάμεσα σε κάθε λέξη. Μετά από αυτό το βήμα η μορφή του κειμένου είναι καθαρή και είναι έτοιμο για να χρησιμοποιηθεί για αναζήτηση.

**Βήμα 3. Ευρετήριο (Indexing):**

Σε αυτό το βήμα δημιουργήσαμε μια ανεστραμμένη δομή δεδομένων ευρετηρίου (inverted index), συγκεκριμένα δημιουργήσαμε ένα λεξικό (dictionary) στο οποίο κάθε λέξη συνδέεται με τα document id από τα έγγραφα στα οποία εμφανίζεται. Η υλοποίηση αυτού του τύπου ευρετηρίου είναι χρήσιμη μόνο στην αναζήτηση με απλή μέθοδο Boolean, εφόσον περιορίζεται από την έλλειψη αποθήκευσης στοιχείων για την συχνότητα της εμφάνισης της κάθε λέξης, όμως εμφανίζει γρήγορα τα αποτελέσματα λόγω της απλότητας του ευρετηρίου καθώς και του αλγορίθμου στον οποίο υλοποιείται.

**Βήμα 4. Μηχανή αναζήτησης (Search Engine):**

**α) Επεξεργασία ερωτήματος (Query Processing):**

Αρχικά, το ερώτημα του χρήστη πρέπει να περάσει μία προεπεξεργασία παρόμοια με αυτή που έχει περάσει το αρχικό κείμενο, δηλαδή το ερώτημα μετατρέπεται σε μικρά γράμματα, επεξεργασία tokenization και stemming πριν γίνει η αναζήτηση. Με την ύπαρξη του ανεστραμμένου ευρετηρίου είμαστε σε θέση να τον υλοποιήσουμε σε μία απλή αναζήτηση τύπου Boolean. Ο αλγόριθμος Boolean εμφανίζει τα αποτελέσματα με την σειρά την οποία τα έψαξε, δηλαδή χωρίς κάποια ταξινόμηση, έχει επίσης την δυνατότητα να αναζητήσει ερωτήματα με λογικούς τελεστές, συγκεκριμένα έχουμε υλοποιήσει το AND, OR και ΝΟΤ.

**β) Κατάταξη αποτελεσμάτων (Ranking):**

Ο πρώτος αλγόριθμος ο οποίος ταξινομεί τα αποτελέσματα τον οποίο υλοποιήσαμε είναι ο TF-IDF (term frequency–inverse document frequency), ο οποίος είναι αλγόριθμος του μοντέλου VSM (Vector Space Model). Ο TF-IDF χρησιμοποιεί δύο μεταβλητές οι οποίες είναι term frequency και inverse document frequency και υπολογίζονται ως εξής:

* Term frequency, tf(t,d):

Είναι η συχνότητα με την οποία εμφανίζεται το term t το οποίο αναζητούμε στο document d και υπολογίζεται από την διαίρεση του αριθμού φορών που εμφανίστηκε στο document το term t με το συνολικό αριθμό μοναδικών λέξεων οι οποίες υπάρχουν στο document.

* Inverse document frequency idf(t,D):

Υπολογίζεται πόσο “σπάνια” είναι μία λέξη t στο σύνολο documents D στα οποία εμφανίζεται

Η τιμή TF-IDF για κάθε term είναι ο πολλαπλασιασμός της τιμής tf με την τιμή idf για το κάθε term TF-IDF(t,d,D)=tf(t,d) • idf(t,D). Ο αλγόριθμος χρησιμοποιείται για ταξινόμηση με βάση την συχνότητα τους στο αρχείο αλλά παίρνει υπόψην την συνολική σπανιότητα της λέξης σε όλα τα κείμενα. Για παράδειγμα, η λέξη “animal” μπορεί να εμφανίζεται συχνά σε ένα dataset το οποίο αφορά ζώα, ενώ σε dataset όπως αυτό που χρησιμοποιήσαμε το οποίο αφορά ταινίες δεν εμφανίζεται συχνά. Στην μία περίπτωση η λέξη εμφανίζεται τακτικά οπότε ο TF-IDF αναγνωρίζει ότι η λέξη δεν είναι σπάνια οπότε την ταξινομεί χαμηλότερα και αντιθέτως στο dataset με τις ταινίες την ταξινομεί ως σπάνια.